Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

1. . (Currently amended) A method of operating a multi-threaded processor comprising:

receiving data ~~specified~~ by execution of a fast-write instruction, the data comprising immediate data supplied with the fast-write instruction, in one of multiple threads processing on the multi-threaded processor, the one of the multiple threads identified by a processing thread number, the fast-write instruction further specifying a register, the register having multiple groups of bits, each group of bits associated with a corresponding thread of the multiple threads processing on the multi-threaded processor;

selecting a group of bits associated with the one of the multiple threads, the group of bits being selected from the multiple groups of bits of the register specified by the fast-write instruction according to the processing thread number; and

loading the data into the bit positions of the selected group of bits of the register.

2. (Original) The method of claim 1 wherein the register is a control and status register (CSR).

3. (Previously presented) The method of claim 2 wherein the control and status register is coupled to a 64-bit wide first-in first-out (FIFO) bus.

4. (Original) The method of claim 3 wherein the FIFO bus interfaces with Media Access Controller (MAC) devices.

5. (Previously presented) The method of claim 1 wherein the data represents hexadecimal mask values 0 to 0x3FF.

6. (Currently amended) The method of claim 1 wherein the multi-threaded processor is a parallel, hardware-based multi-threaded processor comprising a plurality of micro engines, and the one of multiple threads is processed on a micro engine of [[a]] the parallel, hardware-based multi-threaded processor.

7. (Previously presented) The method of claim 1, wherein loading the data comprises:

shifting a first portion of the data by an amount equal to the processing thread number; and

shifting a second portion of the data into bit positions corresponding to a breakpoint (BP) register 2 through a BP register 0.

8. (Previously presented) The method of claim 1 wherein the fast-write instruction comprises a token.

9. (Original) The method of claim 8 wherein the token represents overriding qualifiers.

10. (Original) The method of claim 8 wherein the token is a 32-bit word.

11. (Previously presented) The method of claim 10 wherein a token format comprises:

an OV field in bit 31;

a micro engine (UENG) ADDR field in bits 30:28;

a reserved field in bits 27:16;

an OV field in bit 15;

a fast write data field in bits 14:5;

a reserved field in bits 4:3;

an OV field in bit 2; and

a CTX field in bits 1:0.

12. (Original) The method of claim 11 wherein a micro engine address overrides a default micro engine address if bit 31 is set.

13. (Original) The method of claim 11 wherein bits 30:28 specify a micro engine associated with a control and status register (CSR).

14. (Original) The method of claim 11 wherein bits 27:16 return 0 when read.

15. (Original) The method of claim 11 wherein a micro engine address overrides a default micro engine address if bit 15 is set.

16. (Original) The method of claim 11 wherein bits 14:5 represent valid data to be written to a control and status register (CSR).

17. (Original) The method of claim 11 wherein bits 4:3 return 0 when read.

18. (Original) The method of claim 11 wherein a context (CTX) field overrides a default context if bit 2 is set.

19. (Original) The method of claim 11 wherein bits 1:0 specify a context associated with a control and status register (CSR) reference.

20. (Currently amended) A computer program product, disposed on a computer readable medium, the program comprising a fast-write instruction that causes a computer to:

receive data ~~specified in~~ by execution of the fast-write instruction, the data comprising immediate data supplied with the fast-write instruction, in one of multiple threads processing on a multi-threaded processor, the one of the multiple threads identified by a processing thread number, the fast-write instruction further specifying a register, the register having multiple

groups of bits, each group of bits associated with a corresponding thread of multiple threads processing on the multi-threaded processor;

select a group of bits associated with the one of the multiple threads, the group of bits being selected from the multiple groups of bits of the register specified in the fast-write instruction according to the processing thread number; and

load the data into the bit positions of the selected group of bits of the register.

21. (Previously presented) The computer program product of claim 20 wherein the register is a control and status register (CSR).

22. (Previously presented) The computer program product of claim 21 wherein the control and status register is coupled to a 64-bit wide first-in first-out (FIFO) bus.

23. (Original) The computer program product of claim 22 wherein the FIFO bus interfaces with Media Access Controller (MAC) devices.

24. (Previously presented) The computer program product of claim 20 wherein the data represents hexadecimal mask values 0 to 0x3FF.

25. (Currently amended) The computer program product of claim 20 wherein the multi-threaded processor is a parallel, hardware-based multi-threaded processor comprising a plurality of micro engines, and the one of multiple threads is processed on a micro engine of [[a]] the parallel, hardware-based multi-threaded processor.

26. (Previously presented) The computer program product of claim 20, wherein the fast-write instruction further causes the computer to:

shift a first portion of the data left by an amount equal to the processing thread number; and

shift a second portion of the data into bit positions corresponding to a breakpoint (BP) register 2 through BP register 0.

27. (Previously presented) The computer program product of claim 20, wherein the fast-write instruction comprises a token.

28. (Original) The computer program product of claim 27 wherein the token represents overriding qualifiers.

29. (Original) The computer program product of claim 27 wherein the token is a 32-bit word.

30. (Previously presented) The computer program product of claim 29 wherein a token format comprises:

an OV field in bit 31;

a micro engine (UENG) ADDR field in bits 30:28;

a reserved field in bits 27:16;

an OV field in bit 15;

a fast write data field in bits 14:5;

a reserved field in bits 4:3;

an OV field in bit 2; and

a CTX field in bits 1:0.

31. (Original) The computer program product of claim 30 wherein a micro engine address overrides a default micro engine address if bit 31 is set.

32. (Original) The computer program product of claim 30 wherein bits 30:28 specify a micro engine associated with a control and status register (CSR).

33. (Original) The computer program product of claim 30 wherein bits 27:16 return 0 when read.

34. (Original) The computer program product of claim 30 wherein a micro engine address overrides a default micro engine address if bit 15 is set.

35. (Original) The computer program product of claim 30 wherein bits 14:5 represent valid data to be written to a control and status register (CSR).

36. (Original) The computer program product of claim 30 wherein bits 4:3 return 0 when read.

37. (Previously presented) The computer program product of claim 30, wherein a context (CTX) field overrides a default context if bit 2 is set.

38. (Previously presented) The computer program product of claim 30 wherein bits 1:0 specify a context associated with a control and status register (CSR) reference.